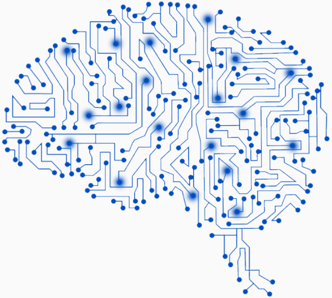
Master 1 Statistique & Data Science, Ingénierie Mathématique

Apprentissage pour l'image
Machine learning for image processing

Presentation of TP1

Emile Pierret

Jeudi 9 mars 2023



Objectives

At the end of the course :

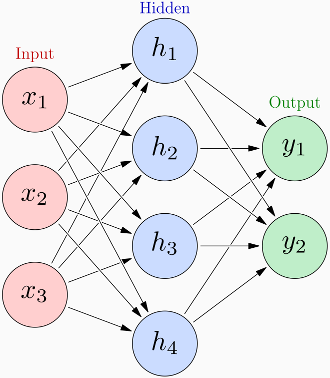
- Know what is a CNN (Convolutional Neural Network)
- Implement the training of a CNN for classification with Pytorch

For this session :

- Train the last layer of a multilayer network for classification with numpy.
- Apply a Stochastic gradient.
- For the moment, no images, no convolutions.

Reminder - What is a multilayer network ?

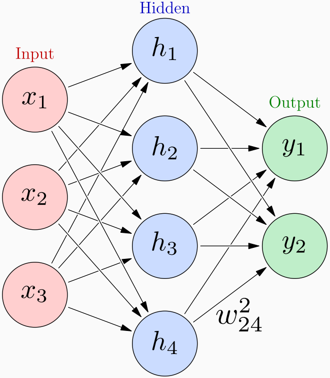
,containsverbatim]



- Inter-connection of several artificial neurons (also called nodes or units).
- Each level in the graph is called a layer:
  - Input layer,
  - Hidden layer(s),
  - Output layer.
- Each neuron in the hidden layers acts as a classifier / feature detector.
- Feedforward NN (no cycle)
  - first and simplest type of NN,
  - information moves in one direction.
- Recurrent NN (with cycle)
  - used for time sequences,
  - such as speech-recognition.

Reminder - What is a multilayer network ?

,containsverbatim]

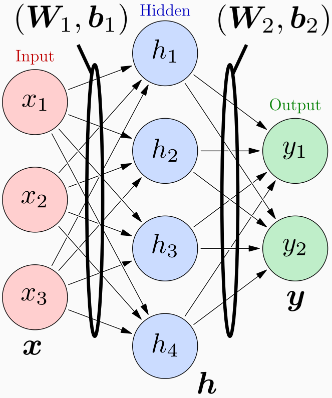


$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$

$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$

Reminder - What is a multilayer network ?

,containsverbatim]



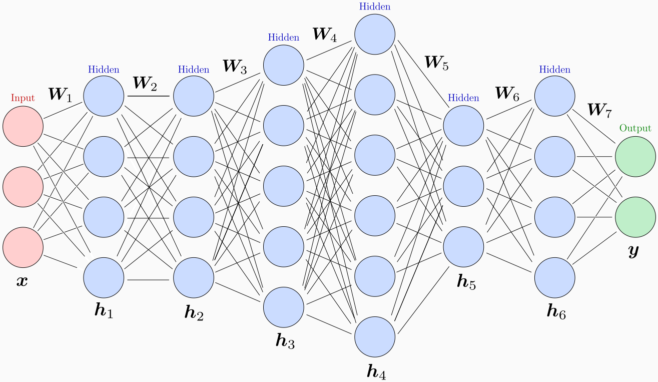
$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$

$$\boldsymbol{h} = g_1\left(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1\right)$$

$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$

$$\boldsymbol{y} = g_2\left(\boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2\right)$$

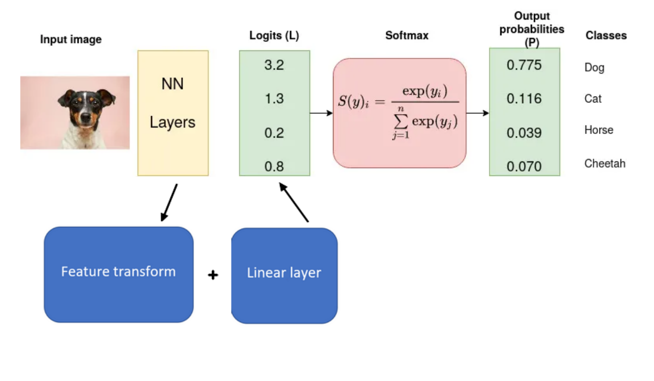
Artificial neural network / Multilayer perceptron



How to train a network ?

1. Consider a training data set
2. Consider a loss $L(W)$
3. Minimize L applying a stochastic gradient

In real-life



In real-life, all the network have to be trained. The feature transform is trained with its linear separation.

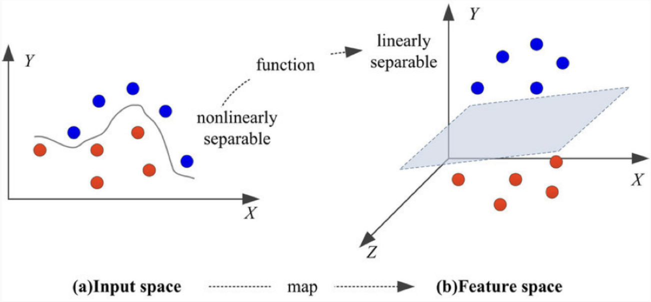
For this session, we will just train the last layer. We suppose that we have a good feature transform to have linearly separable data.

Feature transform

- We apply a feature transform $\varphi : \mathbb{R}^p \to \mathbb{R}^D$ to each $\boldsymbol{x_n}$:

$$\varphi_n = \varphi(\boldsymbol{x_n}), \quad n = 1, \dots, N.$$

- Depending on context it allows to increase $(D > p)$ or decrease $(D < p)$ the dimension in a way to favor class discrimination (e.g. PCA...).
- This is a non linear map that should make the classes linearly separable.



(Source: url)

Recall of the notations

- K classes $C_1, \ldots, C_K$
- $(\boldsymbol{x}_n, t_n)$ the dataset such that $\boldsymbol{x}_n \in C_{t_n}$, $\boldsymbol{x}_n \in \mathbb{R}^d$
- $(\boldsymbol{\varphi}_n, t_n)$ the dataset after feature transform. $\boldsymbol{\varphi}_n \in \mathbb{R}^D$
- We suppose that $P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^K \exp(\boldsymbol{w}_j^T \varphi)}$
- $\boldsymbol{y}_k(\varphi_n) = \frac{\exp(w_k^T \varphi_n)}{\sum_{j=1}^K \exp(w_j^T \varphi_n)}$

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !
- More precisely, we would like to maximize :

$$\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)$$

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !
- More precisely, we would like to maximize :

$$\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)$$

or minimize

$$-\log\left(\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)\right) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !
- More precisely, we would like to maximize :

$$\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)$$

or minimize

$$-\log\left(\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)\right) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

- How to minimize it ?

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^K \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !
- More precisely, we would like to maximize :

$$\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W \right)$$

  or minimize

$$-\log\left(\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W \right) \right) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n) \right)$$

- How to minimize it ?
- With the gradient descent !

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !
- More precisely, we would like to maximize :

$$\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)$$

or minimize

$$-\log\left(\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)\right) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

- How to minimize it ?
- With the gradient descent !
- What do we need ?

The feature transform being done we suppose that :

$$P(C_k|\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}$$

How to learn the weights $(\boldsymbol{w}_j)_{1 \leqslant j \leqslant K}$?

- The loss is the cross entropy, deriving from ?
- The maximization of the log-likelihood !
- More precisely, we would like to maximize :

$$\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)$$

  or minimize

$$-\log\left(\mathbb{P}\left(\bigcap_{n=1}^{N} \boldsymbol{\varphi_n} \in C_{t_n} \mid W\right)\right) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

- How to minimize it ?
- With the gradient descent !
- What do we need ?
- The gradient !

Computation of the gradient

$$L(\boldsymbol{W}) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

- Linear part: Partial gradient with respect to column w_ℓ, $\ell \in \{1, \ldots, K\}$:

$$\nabla_{w_{t_n}} \left[w_{t_n}^T \varphi_n\right] = \varphi_n$$
$$\nabla_{w_\ell} \left[w_{t_n}^T \varphi_n\right] = 0 \text{ if } \ell \neq t_n$$

- Partial gradient of $\nabla_{\boldsymbol{w}_\ell} \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$?

$$\nabla_{\boldsymbol{w}_\ell} \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right) = \nabla_{\boldsymbol{w}_\ell} \log\Big(\exp(\boldsymbol{w}_\ell^T \varphi_n) + \text{constant}\Big)$$
$$=?$$

A point on the gradients (reminder of the course "optimization"?)

Recall that for $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$, for $h \in \mathbb{R}^n$,

$$d(g \circ f)(x)(h) = dg(f(x))(df(x)(h)).$$

Here,

$$\begin{aligned} d(g \circ f)(x)(h_1) &= \nabla(g \circ f)(x)^T h_1 && \text{for } h_1 \in \mathbb{R}^n \\ dg(x)(h_2) &= g'(x)h_2 && \text{for } h_2 \in \mathbb{R} \\ df(x)(h_3) &= \nabla f(x)^T h_3 && \text{for } h_3 \in \mathbb{R}^n \end{aligned}$$

Consequently,

$$\nabla(g \circ f)(x)^T h = g'(f(x))\nabla f(x)^T h$$

So,

$$\nabla(g \circ f)(x) = g'(f(x))\nabla f(x)$$

Multivariate logistic regression

Gradient of log-likelihood:

Recall that for $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$

$$\nabla (g \circ f)(x) = g'(f(x)) \nabla f(x).$$

Here,

$$g(t) = \log(\exp(t) + c) \qquad g'(t) = \frac{\exp(t)}{\exp(t) + c}$$

$$f(\boldsymbol{w}_\ell) = \boldsymbol{w}_\ell^T \varphi_n \qquad \nabla f(\boldsymbol{w}_\ell) = \varphi_n.$$

So,

$$\nabla_{\boldsymbol{w}_\ell} \log \left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n) \right) = \frac{\exp(\boldsymbol{w}_\ell^T \varphi_n)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)} \varphi_n = \boldsymbol{y}_\ell(\varphi_n) \varphi_n$$

since

$$\boldsymbol{y}_k(\varphi) = \frac{\exp(\boldsymbol{w}_k^T \varphi)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi)}, \quad k = 1, \ldots, K.$$

Multivariate logistic regression

Gradient of log-likelihood:

$$L(\boldsymbol{W}) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

- For each column $w_\ell \in \mathbb{R}^D$ of $\boldsymbol{W}$, $\ell \in \{1, \ldots, K\}$,

$$\nabla_{\boldsymbol{w}_\ell} L(\boldsymbol{W}) = \sum_{n=1}^{N} (\delta_{\ell,t_n} - \boldsymbol{y}_{t_n}(\varphi_n))\varphi_n \in \mathbb{R}^D$$

- OK with intuition ?

One word on the stochastic gradient descent

We would like to minimize w.r.t $\boldsymbol{W}$,

$$L\left(\boldsymbol{W}, (\boldsymbol{\varphi}_n, t_n)_{1 \leqslant n \leqslant N}\right) = -\sum_{n=1}^{N} w_{t_n}^T \varphi_n + \log\left(\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \varphi_n)\right)$$

$$= \sum_{n=1}^{N} \ell\left(\boldsymbol{W}, \boldsymbol{\varphi}_n, t_n\right)$$

Algorithm: (stochastic) gradient descent for $L(\boldsymbol{w})$

- Initialize $\boldsymbol{W}$ randomly
- Repeat until convergence
  - For all $(\boldsymbol{\varphi}_n, t_n)$, $1 \leqslant n \leqslant N$
    - For each $\boldsymbol{w}_k$, $1 \leqslant k \leqslant K$
      Update: $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_k - \gamma \nabla_{\boldsymbol{w}_k} \ell\left(\boldsymbol{W}, \boldsymbol{\varphi}_n, t_n\right)$
- γ is called the learning rate.

Comparaison with constant step gradient descent

Algorithm: Constant step gradient descent for $f(\boldsymbol{x})$

- Initialize $\boldsymbol{x}$ randomly
- Repeat until convergence
  - Update $\boldsymbol{x} \leftarrow \boldsymbol{x} - \gamma \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$
- Why "stochastic" previously ?

Comparaison with constant step gradient descent

Algorithm: Constant step gradient descent for $f(\boldsymbol{x})$

- Initialize $\boldsymbol{x}$ randomly
- Repeat until convergence
  - Update $\boldsymbol{x} \leftarrow \boldsymbol{x} - \gamma \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$

- Why "stochastic" previously ?
- Because we suppose that the data follow a distribution $\mathbb{P}_{\text{data}}$ and we would like to minimize

$$\mathbb{P}_{\text{data}}(\boldsymbol{\varphi} \in C_k \mid W) = \mathbb{E}_{\mathbb{P}_{\text{data}}}\left[\mathbf{1}_{\varphi \in \mathcal{C}_k} \mid W\right]$$

  w.r.t $\boldsymbol{W}$
- In our context, we suppose that our dataset are samples that represents the distribution of the distribution of the features.

Theoretical stochastic gradient descent

We would like to minimize :

$$L(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{u}}(\ell(\boldsymbol{W}, \boldsymbol{u}))$$

Algorithm: (stochastic) gradient descent for $L(\boldsymbol{W})$

- Initialize $\boldsymbol{W}$ randomly
- Repeat until convergence
    - Sample $\boldsymbol{u} \sim \mathbb{P}_{\boldsymbol{u}}$ independent from the previous samples
      Update : $\boldsymbol{W} \leftarrow \boldsymbol{W} - \gamma \nabla_{\boldsymbol{W}} \ell(\boldsymbol{W}, \boldsymbol{u})$
- In practice, γ is not constant.
- There exists theoretical results of convergence.

What is it necessary to code ?

1. A toy dataset
2. The gradient of the loss
3. The gradient descent
4. An evaluation of the outputs.